Hochschule Konstanz
Fakultät Informatik

Machine Learning:: Recap Statistic / Exploratory Data Analysis

Dozenten: Prof. Dr. M. O. Franz, Prof. Dr. O. Dürr

May change until lecture

Remark: Much of the material have been developed together with Beate Sick (ZHAW/Uni Zurich).

Overview

- Notes on Python and literate programming
- The need for statistics in ML
- Recap statistics
  - Sampling
  - Parameter Estimation
  - Quality of parameter estimation
  - Maximum Likelihood principle
- Exploratory data analysis
  - Empirical correlation

Note on Python

- Reproducability requires a widely adopted scripting language with easy access to most scientific software libraries.
- Compare to Matlab: no license costs, indexing and memory model conforms to most programming languages (index starts with 0, row major order, . . .)
- Simple, intuitive code, easy to learn (Rapid Prototyping)
- Modular and object oriented programming, simple management and installation of packages, good integration of documentation and unit tests.
- Large community, direct integration of most numerical C and Fortran libraries which can be executed in parallel therads or on GPUs.
- Disadvantages: heavy version dependence (particularly version 2.x vs. 3.x); as interpreted language, pure Python code is considerably slower than compiler-based languages.

Python software stack

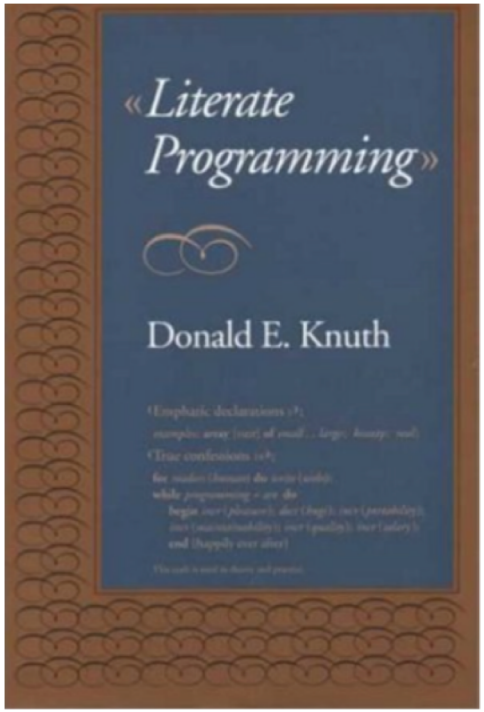
- Numpy - numerical Python: based on the same numerical C/Fortran libraries as Matlab (LAPACK, BLAS, ATLAS), for very eicient vector, matrix and high-dimensional array operations.
- Scipy - scientific Python: based on Numpy, for numerical integration, diferential equations, FFT, signal and image processing, interpolation, optimization, statistics, linear algebra, ...
- Matplotlib - mathematical ploting library: 2D and 3D graphics and diagrams (similar to Matlab or Mathematica).
- Pandas - Python Data Analysis Library: eicient structures for data analysis, in particular for tables and time series.
- Important for this course: Scikit-learn - toolbox for machine learning

- Later:
  - Keras a deep learning framework

Ipython and literate programming

- ”Literate Programming”: Explaining program logic in natural language, interspersed by ”code snippets”.
- The code snippets can be arranged in arbitrary order. The literate programming system automatically assembles them in an executable order if desired (tangle).
- At the same time, the system is able to automatically generate a readable documentation with index, references, etc. (weave).

- Motivation: Better programs and documentation since the author explicitly explains his thoughts during the writing of the program. In our case: reproducible research.



Quelle: Wikipedia

Some example data sets

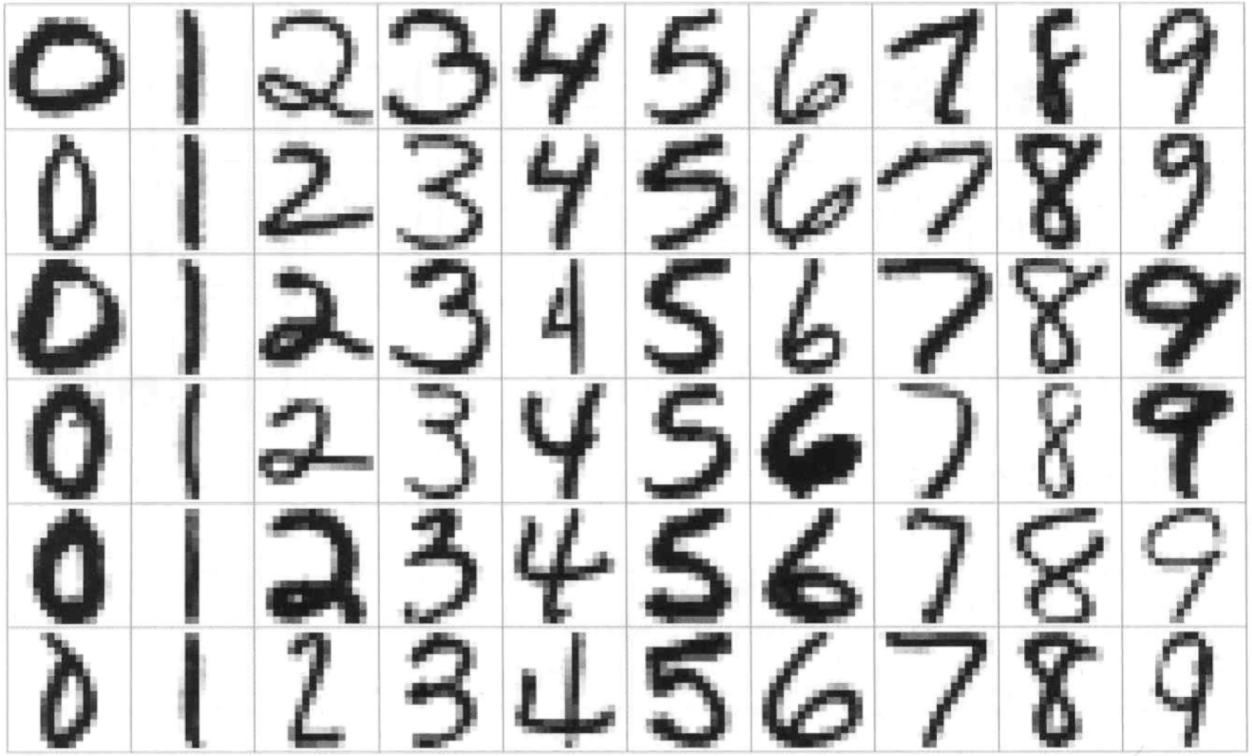
Example1: *Prostate Cancer*

Data from a medical study (97 participants, Stamey et a., 1989) on the relation of various clinical parameters (see below) shortly before an operative prostatectomy.

- lpsa: log level of prostate specific antigen
- lcavol: log cancer volume
- lweight: log prostate weight
- lbph: log of benign prostatic hyperplasia amount
- svi: seminal vesicle invasion (categorical, 0 or 1)
- lcp: log of capsular penetration
- gleason: Gleason score (categorical, 6,7,8,9)
- pgg45: percent of Gleason score 4 and 5

Goal: Prediction of one variable from the others (i.e. classification and regression).

Example 2: Hand-written digit recognition

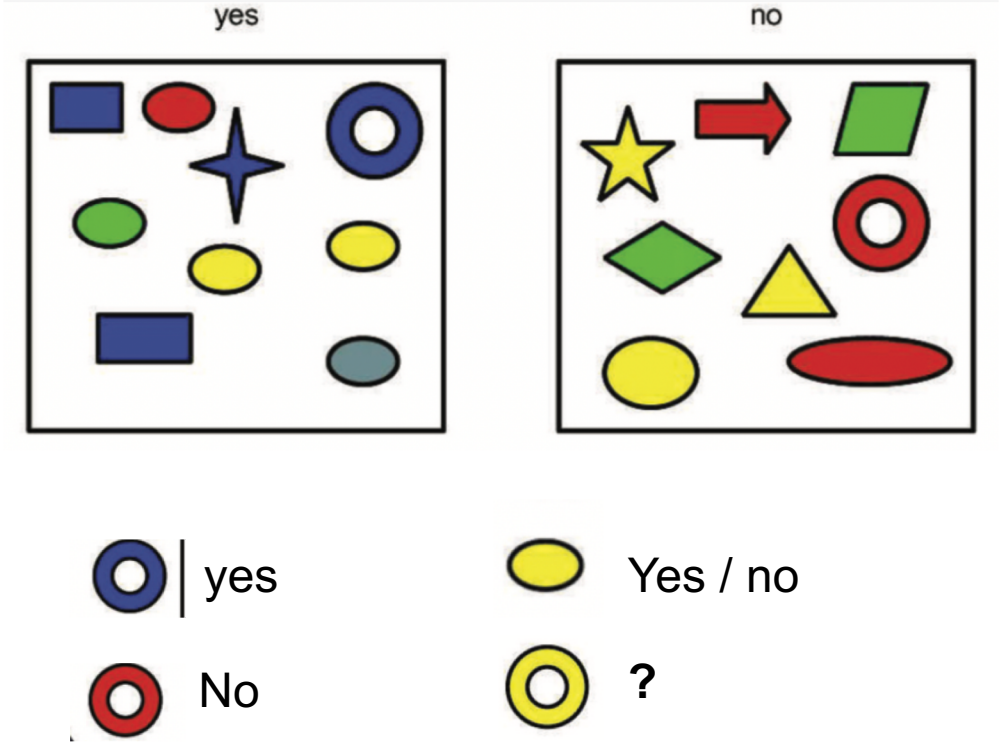


Data type: 16 x 16 grayvalue images (0..255) of scanned, binarized hand-written zip codes from US letters.

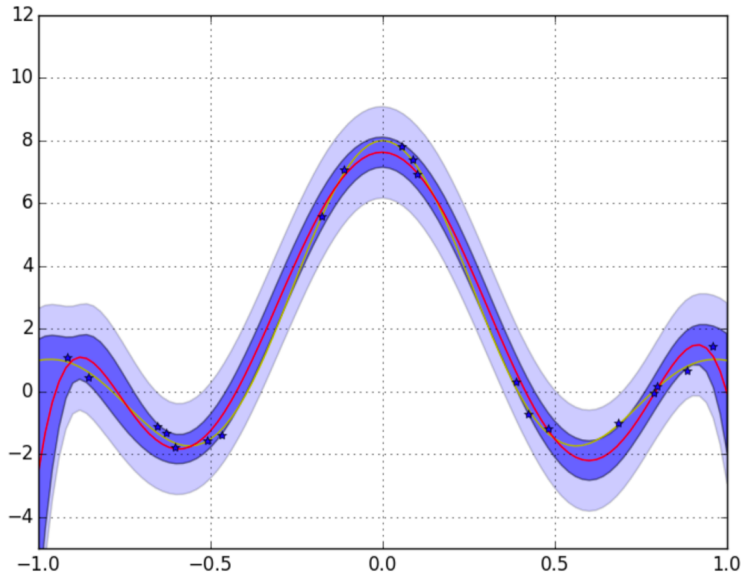
Goal: classification of the images as belonging to the classes 0 to 9 or "don't know".

Need for probabilistic prediction

Classification



Regression



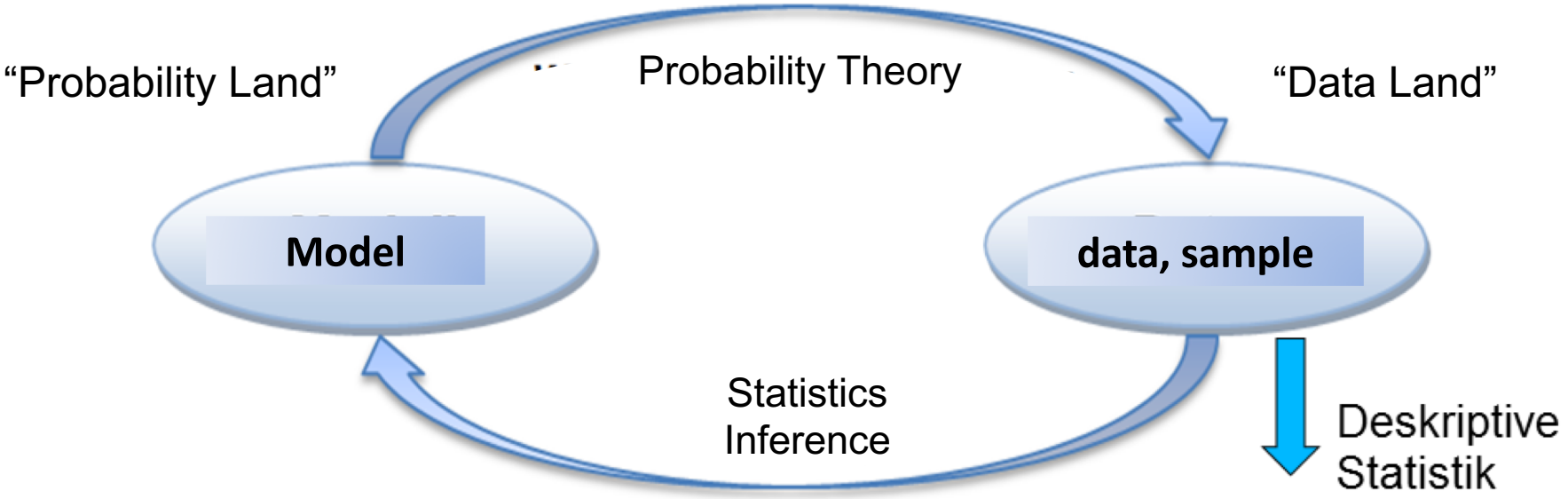
Data is noisy ➔ need statistical description
In many ML applications probabilistic forecasts are needed.

Adapted from Murphy 2012

Short Recap of Statistics

Eat your own dogfood: this section is accompanied by the ipython notebook
https://github.com/ioskn/mldl_htwg/blob/master/intro/intro.ipynb

Statistics connects data with models



describe data
Visualize & Summarize

Statistical inference

- Parameter estimation
  - Regression
  - Classification
- Confidence intervals
- Tests

Probability: model → data

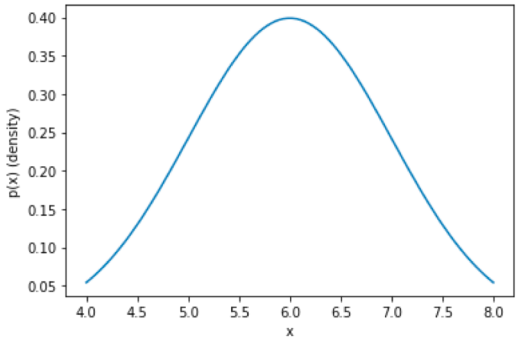
- Probability Theory
  - A coin is thrown $\boldsymbol{n = 20}$ times, the probability for one head is $\boldsymbol{\pi = 0.2}$ how probable is it to have k heads? T**he parameters are known**

$$k \sim Binom(\mathrm{n} = 20, \pi = 0.2)$$

  - You drive 100 km, how many liters of gas x do you need? We assume that the data **is sampled/drawn from or generated by a** normal distribution with $\mu = 6, \sigma^2 = 1$ (**the parameters are known**)

$$x \sim N(\mu = 6, \sigma^2 = 1)$$

  - Properties of pmf, pdf [**Blackboard**]
    - sum / integrate to one
    - Expectation is center of mass



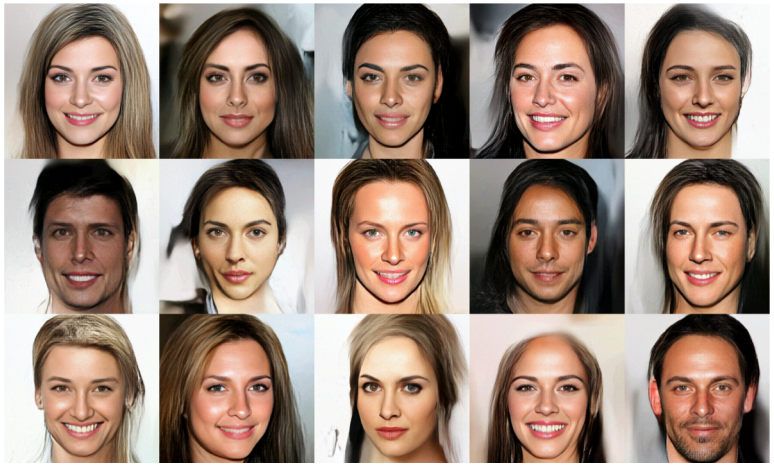
We here drop the distinction between X (RV) and x (sample)

Probability: model → data (sampling)

- Data varies
  - ipython example of car [notebook]
    - np.random.normal(loc=6,scale=1,size=10)
  - Model with some parameters (knobs are μ and σ)

$x \sim$ 

  - x can also be high dimensional data (256x256x3)



15 different x sampled from a deep learning model called glow (more than 100 mio. Parameters, 800 MB)

These are not real people!

For glow see: arXiv:1807.03039

Inference: data → model (training)

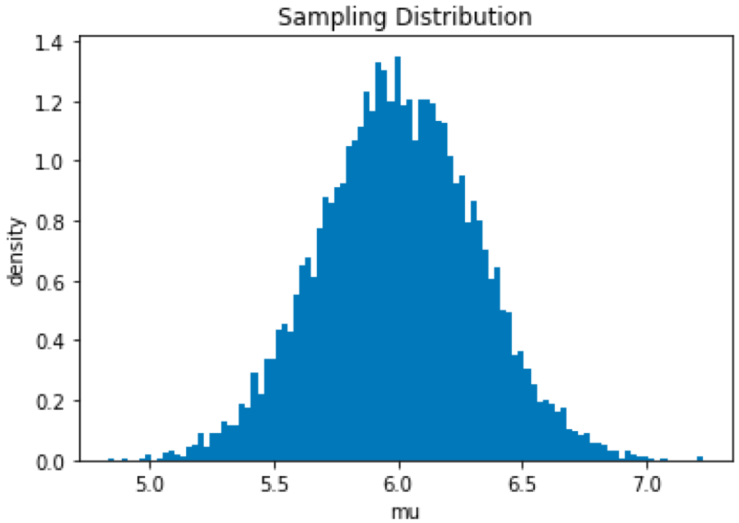
- Parameter Estimation:
  - A coin is thrown n=20 times, you observe 12 heads, what is your best guess for pi?
  - A car is driven n=10 times 100 km you observe the following 10 values:
    - [6.27045683, 5.94976189, 5.76105195, 5.09243634, 5.42322867, 6.75539123, 6.50091719, 5.02244476, 6.09933231, 6.75138712]
  - **You assume** it is a Normal Distribution

$$x \sim N(\mu, \sigma^2)$$

  - What is your best guess μ and σ ? Any idea?
  - Cook book
    - Try to connect probability land and data land
      - expectation < == > mean
      - variance < == > (empirical variance)
    - **Blackboard**

Goodness of estimation

- Is taking the mean of the data good estimate for the expectation?



- The expectation of estimate should be the real value (**unbiasedness**)
- Sample size $n \to \infty$ estimate should be true value (**consistent**)
  - a bit complicated to formulate that correctly (we don't care here)
- The spread of the sampling distribution should be small (efficiency)
- **Notebook**

Parameter Optimization with Max Likelihood Optimization

Die Maximum-Likelihood (ML) Methode

Sie haben einen Würfel mit z.B. l=2 roten Seiten.

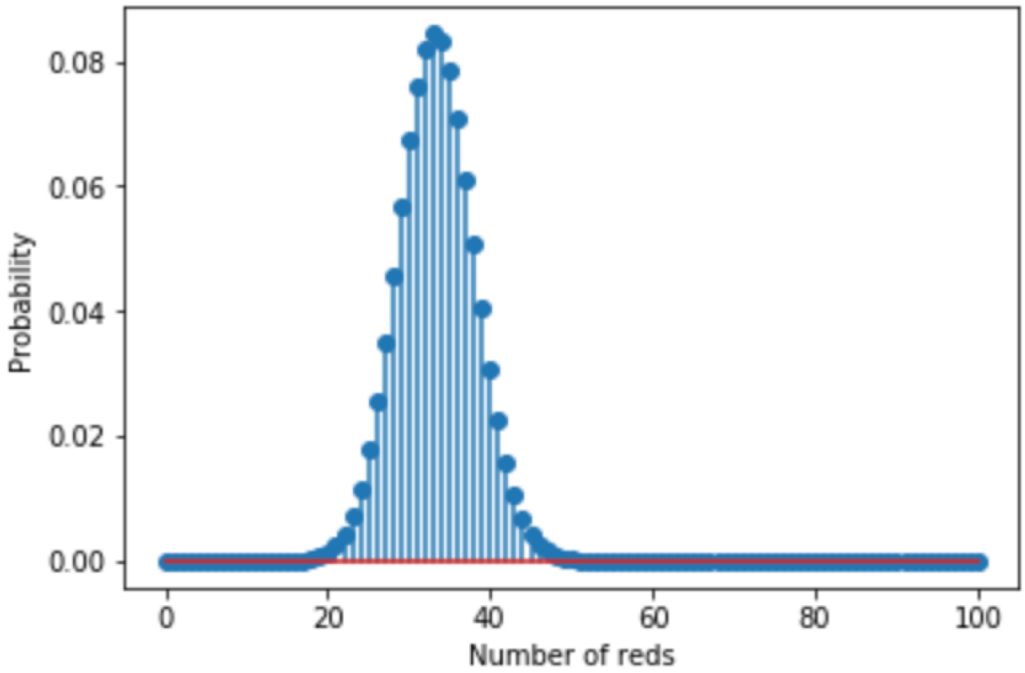
Wie Wahrscheinlich ist es, dass Sie bei 100 Würfen:

- *Keine rote Seite gewürfelt haben*
- *34 rote Seiten gewürfelt haben*
- *99 rote Seiten gewürfelt haben*

$$P(X = k \mid p) = \begin{pmatrix} 100 \\ k \end{pmatrix} \cdot p^k \cdot (1-p)^{100-k}$$

$$p = \frac{2}{6}$$

Ergebniss




```
from scipy.stats import binom
reds = np.asarray(np.linspace(0,100,101), dtype='int') #The numbers 0 to 10 as integers
p_reds = binom.pmf(k=reds, n=100, p=2/6) #The probability of 0,1,2...,throws
plt.stem(reds, p_reds)
plt.xlabel('Number of reds')
plt.ylabel('Probability')
```

Umdrehen der Argumentation

- Wir haben 34 mal rot gesehen, welche Wert des Parameters erklärt die Daten am Besten (hat die grösste Wahrscheinlichkeit)
-

Die Maximum-Likelihood (ML) Methode

Sie haben einen Würfel mit z.B. l=2 roten Seiten.

Wie Wahrscheinlich ist es, dass Sie bei 100 Würfen:

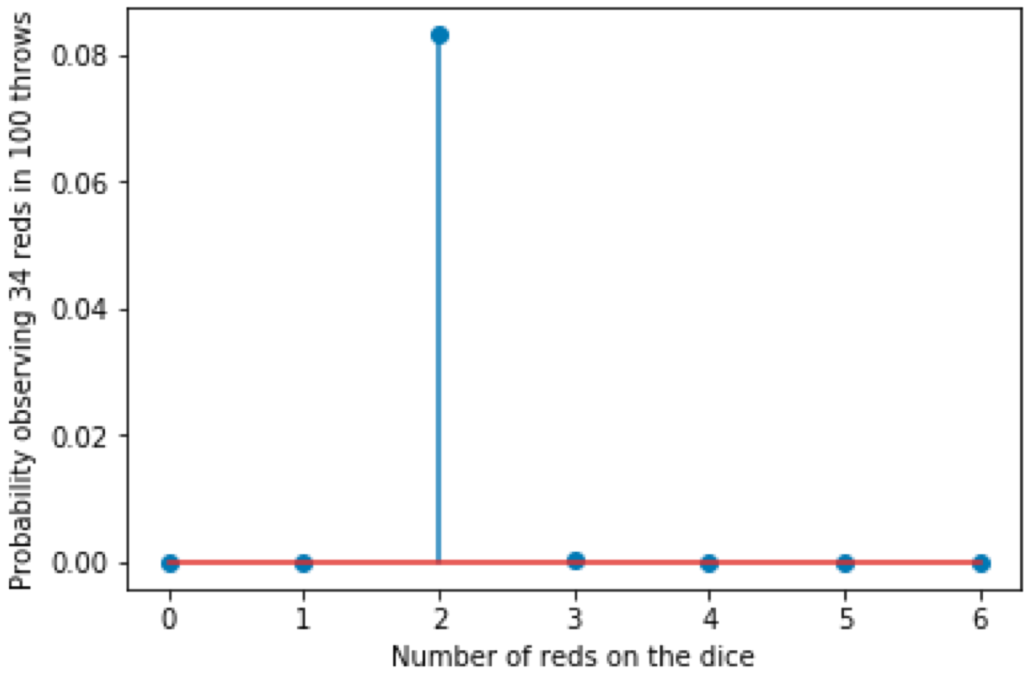
- *Keine rote Seite gewürfelt haben*
- *34 rote Seiten gewürfelt haben*
- *99 rote Seiten gewürfelt haben*

$$P(X = k \mid p) = \binom{100}{k} \cdot p^k \cdot (1-p)^{100-k}$$

$$p = \frac{2}{6}$$

In R ausprobieren.

Varying p



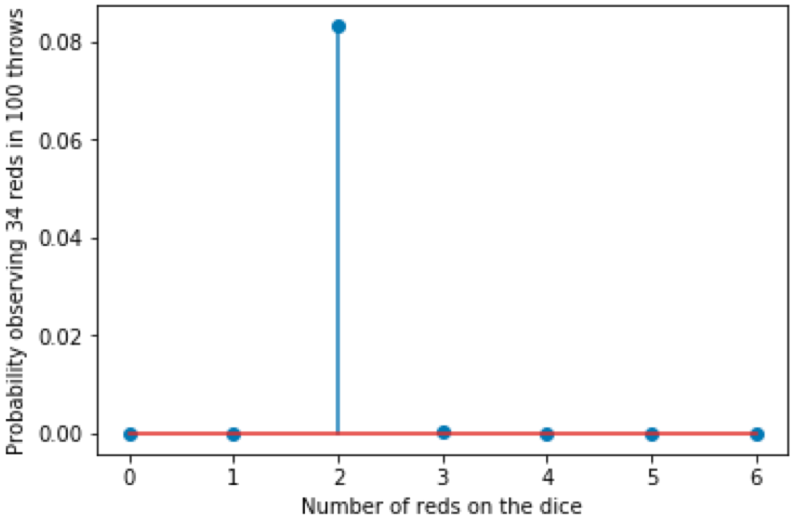

```
# Solution
from scipy.stats import binom
num_reds = np.asarray(np.linspace(0,6,7), dtype='int')
p_num_dollar = binom.pmf(k=34, n=100, p=num_reds/6)
```

ML-Methode Was haben wir getan?

$$L(X \mid \theta) = P(X = k \mid p) = \begin{pmatrix} 100 \\ k \end{pmatrix} \cdot p^k \cdot (1-p)^{100-k}$$

Fest

Der Parameter Variiert



ML-Methode Was haben wir getan?

$$L(X \mid \theta) = P(X = k \mid p) = \begin{pmatrix} 100 \\ k \end{pmatrix} \cdot p^k \cdot (1-p)^{100-k}$$

Fest

Der Parameter
Variiert

Wir haben das p so gewählt, dass es dem Maximum der Wahrscheinlichkeit genügt.

Anmerkung: P(X=k|p) (als Funktion von p) ist keine Wahrscheinlichkeit im engerem Sinne, denn z.B.

`sum(dbinom(x = 34, prob = c(1:6)/6, size = 100))=0.083`

Maximum Likelihood
(one of the most beautiful ideas in statistics)

Ronald Fisher in 1913
Also used before by
Gauss, Laplace

Likelihood / “probability“
(often known)

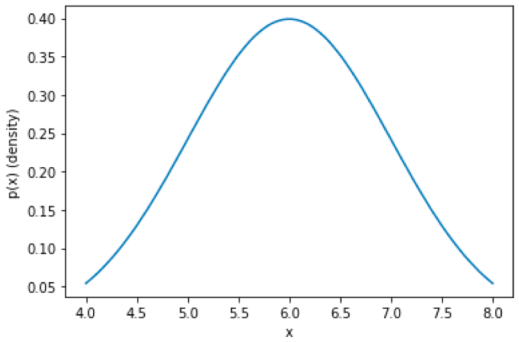
$$M(\theta) \longrightarrow \text{Data}$$

Tune the parameter(s) θ of the model M so that (observed) data is most likely

Introducing features: $p(x) \rightarrow p(y|x)$

- In ML, we mostly have features x (not a random variable)
  - E.g. x=100 km in the car example
  - Instead of x, we now predict the random variable y given x
  - Most ML can be formulated in the statistical language

$p(x)$



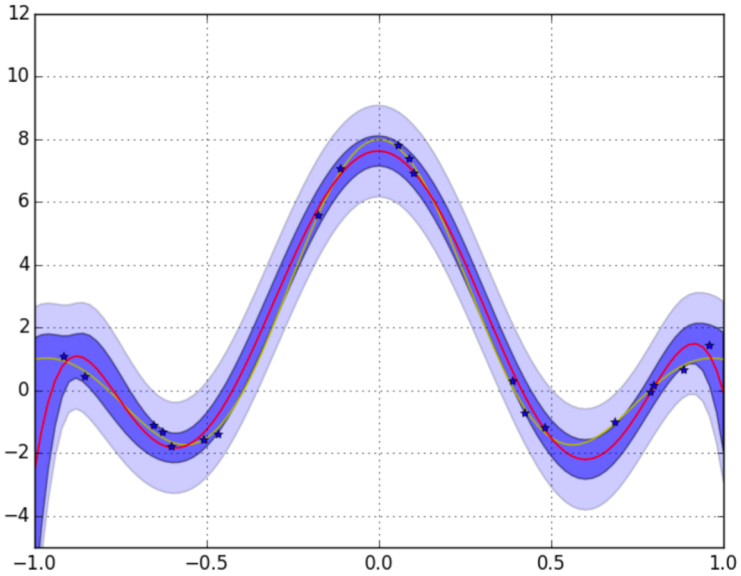
x

$x \rightarrow$

→y

$p(y|x)$

Regression



x

For the regression problem, we have at every x a distribution (hard to draw).

Preprocessing and EDA

Preprocessing components

- **Raw data:** non-manipulated data to which no software has been applied.
- **Processed data:**
  - Erroneous measurements are removed.
  - Data are arranged in tables, i.e., one variable per column, one observation per row, readable variable names in the first row.
  - In multiple tables there should be a column that links all tables together. Only one table per file.
- **Code book/README:** Information in text oder Markup file about
  - Variables (e.g. units)
  - Type of statistics applied
  - Experimental design of the study, references, URLs, etc.
- **Instruction list:** a script with raw data as input and processed data as output, no parameters.

Requirement for Data in ML

1. The input variables (or their combinations) should have a statistical dependency to the output variables that are to be predicted.
2. The data should be **scaled** such as to avoid numerical problems during processing and to allow each variable to influence the outcome.
3. The input variables should be as statistically independent as possible such that each variable provides additional information about the outputs.
4. The number of variables should be as small as possible since machine learning methods work better for low-dimensional data

It is therefore a good idea to do an explorative data analysis before applying machine learning algorithms to check the statistical dependencies, data types and scaling of the data.

Note: 3 and 4 does not apply for Deep Learning

A simple measure of statistical dependency: correlation

- Definition of correlation (not to be mixed up with correlation coefficient)

$$C_{xy} = E(x \cdot y) = \sum_x \sum_y x \cdot y \, p(x,y) \qquad \text{discrete}$$

$$C_{xy} = E(x \cdot y) = \int \int x \cdot y \, p(x,y) \, dx \, dy \qquad \text{continious}$$

- Estimation (E→MW) of the correlation coefficient on N data points

$$\widehat{C_{xy}} = MW(x \cdot y) = \frac{1}{N} \sum_1^N x_i \cdot y_i$$

- If x and y are statistically dependent they typically are positive or negative at the same time (they are covariant). Consequently, their product is oen large and positive which results in a positive correlation.
- If the variables vary independently of each other their product typically remains small since x will be oen near 0 when y is large and vice versa. Additionally, positive and negative products will cancel each other in the sum.

Covariance: a dependency measure independent of the mean value

For statistically independent variables, we have $p(x, y) = p(x) \cdot p(y)$. Their correlation is in this case

$$\begin{aligned} C_{xy} &= \sum_{x_i \in \mathcal{X}} \sum_{y_j \in \mathcal{Y}} xy\, p(x, y) = \sum_{x_i \in \mathcal{X}} \sum_{y_j \in \mathcal{Y}} xy\, p(x) p(y) \\ &= \left(\sum_{x_i \in \mathcal{X}} x\, p(x) \right) \cdot \left(\sum_{y_j \in \mathcal{Y}} y\, p(y) \right) = \mu_x \cdot \mu_y \end{aligned}$$

Generally, the correlation is higher when the variable means are higher, independently of whether they are covariant or not. A better measure of statistical dependency is therefore the **covariance** of x and y

$$\sigma_{xy} = \mathcal{E}[(x - \mu_x)(y - \mu_y)] = \sum_{x_i \in \mathcal{X}} \sum_{y_j \in \mathcal{Y}} (x - \mu_x)(y - \mu_y)\, p(x, y),$$

which is indeed 0 for statistically independent variables.

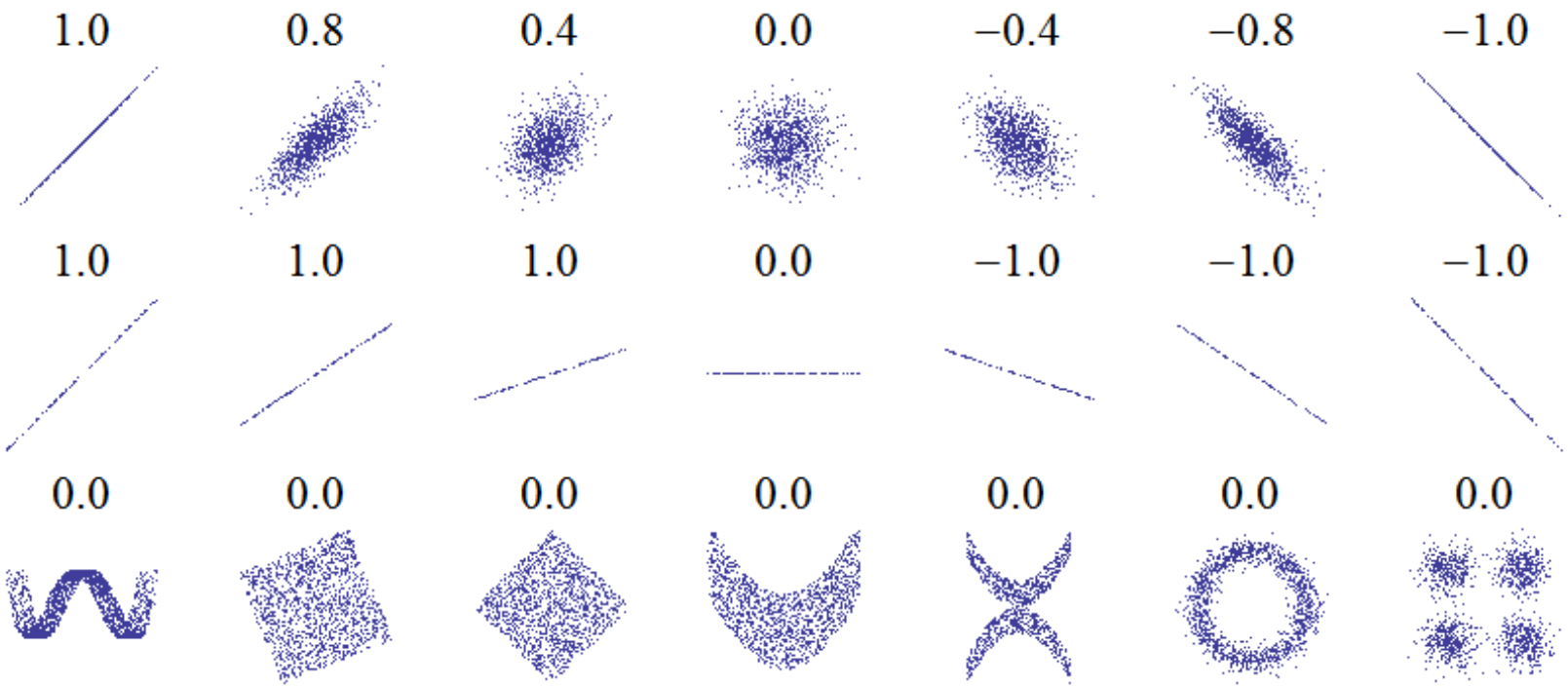
Correlation coefficient after Bravais-Pearson (1)

Next problem: the higher the standard deviation of the variables, the higher will be their covariance. We need to further correct our dependency measure by dividing through the standard deviation of each variable which gives the **correlation coefficient** after Bravais-Pearson:

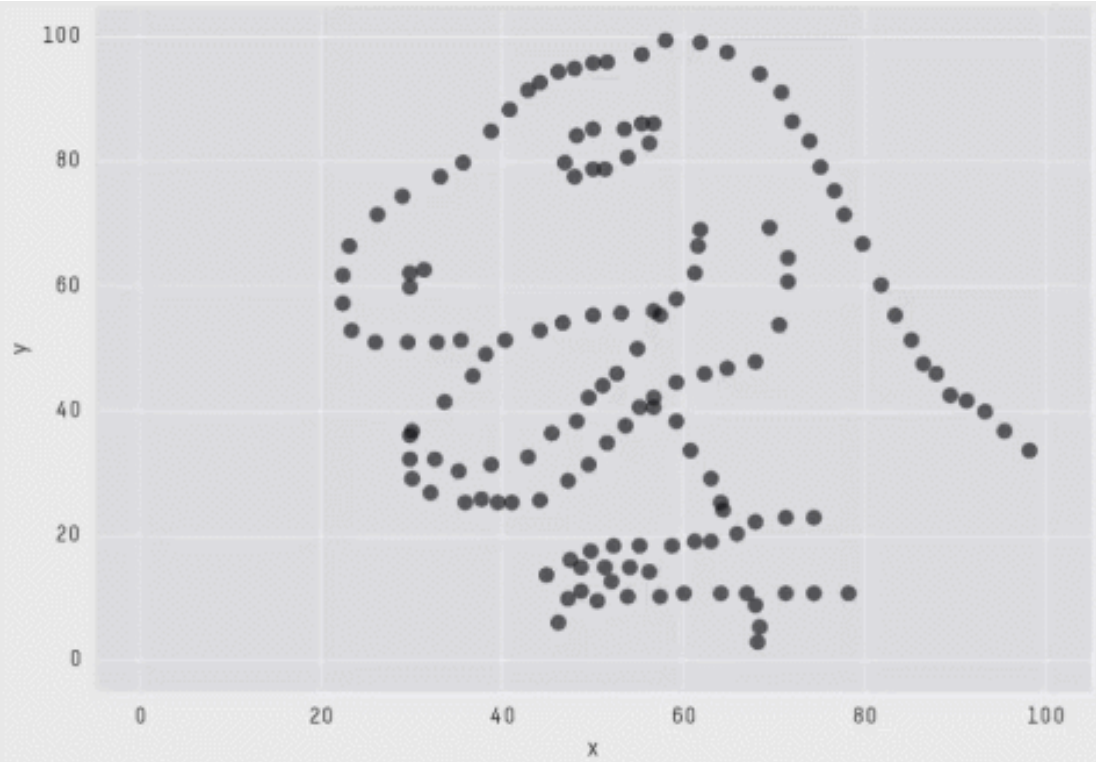
$$r_{xy} = \frac{\sigma_{xy}}{\sigma_x \sigma_y}$$

- r_{xy} near 1 means a strong linear dependency between x and y where y is typically large when also x is large (**positive correlation**).
- r_{xy} near 0 a weak or no linear dependency between x and y (x and y are **uncorrelated**.).
- r_{xy} near -1 means a strong linear dependency between x and y where y is typically large and positive when x is large and negative (**negative correlation**).

Scaerplots for various empirical correlation coefficients



Correlation only capture linear dependencies



X Mean: 54.2659224
Y Mean: 47.8313999
X SD : 16.7649829
Y SD : 26.9342120
Corr. : -0.0642526

MW 0, STD 1, Corr 0
https://www.r-bloggers.com/the-datasaurus-dozen/

Limits of correlation analysis

- Correlations are not the only form of statistical dependency. It often happens that two variables are uncorrelated but still are dependent on each other.
  - Use scatterplots!

- Estimates of correlation coefficients are quite unstable, especially for small sample sizes. In addition, the correlation coeicient strongly reacts to outliers. There are statistical tests for checking whether the estimated coefficient is significantly above 0.

- In multidimensional problems, the data cannot be displayed in a single scatterplot. In order to still get some idea about the correlational structure of our data, we display the scaerplot of each pair of variables in the form of a scatterplot matrix